TrI Toolbox

User Manual

Olga F. Lazareva

David P. Goodman

Department of Psychology

Drake University

2014, ver. 1.0

Table of Contents

1.0 BACKGROUND

1.1. TYPICAL N-TERM TRANSITIVE INFERENCE TASK

Transitive inference (TI) is a deductive reasoning ability to derive a new relationship, *a* ***R*** *c* from the already known relationships, *a* ***R*** *b* and *b* ***R*** *c* (Johnson-Laird, 1999). For example, knowing that *Julia* ***is faster than*** *Jim* and *Jim* ***is faster than*** *Juan*, a transitively competent individual ought to infer that *Julia* ***is faster than*** *Juan*.

In a traditional nonverbal TI task termed *n-series task*, a subject is presented with 4 (or more) overlapping pairs of stimuli: A+ B-, B+ C-, C+ D-, and D+ E- (where A, B, C, etc. indicate the stimuli, pluses indicate reinforcement, and minuses indicate non-reinforcement). In this 5-term task, the critical testing pair is the pair BD as it does not contain end-anchor stimuli A (always reinforced) or E (never reinforced). The ability to select a stimulus B in the pair BD is often interpreted as indicative of transitive inference. Additional training stimuli (e.g., F, G, or more) allow more than one testing pair that does not involve end-anchor stimuli (see Lazareva, 2012, for more details). The TrI Toolbox is designed to provide predictions for the *n-term series task* of any length.

1.2. REINFORCEMENT-BASED MODELS OF TRANSITIVE INFERENCE

Although many reinforcement-based TI models exist (see Lazareva, 2012; Vasconcelos, 2008, for overviews), most algebraically based models share the same basic features. According to those models, during standard TI training the associative values of the training stimuli begin to form an ordered series of associative values: A > B > C > D > E. Thus, at the end of the training a subject could choose stimulus B over stimulus D in the critical pair BD based on a direct comparison of their relative associative values instead of using inferential-like processes.

Reinforcement-based models that are able to capture most of the standard behavioral phenomena include context-dependent configural associative values in addition to context-independent direct associative values (see Wynne, 1995, 1997, 1998). TrI toolbox implements both Siemann-Delius (Siemann & Delius, 1998) and Wynne (1998) configural models. In addition, TrI toolbox provides an opportunity to apply bidirectional value transfer to both models. All equations used in these calculations are provided in Appendix.

During simulations, the TrI Toolbox tests all possible combinations of free-fitting parameters (technically known as exhaustive search of the solution space) to find the best possible combination that *fits terminal performance to the training pairs*. Then, using associative values accrued with these values of the free-fitting parameters and a subject-specific training sequence, the TrI toolbox *predicts testing performance for all possible testing pairs*. In other words, the TrI Toolbox does not attempt to directly fit subjects' testing performance; instead, it fits training data and uses the results of this fit to produce predictions for behavior during testing.

1.3. RECOMMENDED APPROACH TO SIMULATING BEHAVIORAL DATA

Reinforcement-based TI models are highly sensitive to composition of the training session, order in which different training pairs are introduced, number and distribution of correction trials, number and distribution of errors, and similar procedural variables. Because of that, we do not recommend deriving predictions on the basis of "average" sequence of trials. Instead, we recommend the following approach to simulations:

1. **Conducting simulations separately for each subject**, on the basis of their individual trial sequence.

2. **Recording correction trials during experimentation and using them in simulations.** Leaving correction trials out of simulation can dramatically change their outcome (Wynne, 1995). If the correction trials in your experiment were not recorded, we recommend adding 1-3 correction trials at random after each incorrect choice. This is not an ideal solution as it will ignore the likely uneven distribution of correction trials across training pairs (e.g., the errors on inner pairs such as B+ D- or C+ D- are likely to be followed by more correction trials than the errors on the end pairs such as A+ B- or D+ E-), but it is better than leaving correction trials out of simulation completely.

3. **Using appropriate terminal performance as the target performance in simulations.** Terminal performance is the accuracy to each of the training pairs that the model is attempting to fit. Ideally, this performance should be measured at the end of training and prior to the test; in many experiments, this could be the last session(s) or block(s) during which a subject is required to achieve a certain criterion in order to proceed to the test. Some experiments use a fixed number of trials after which a subject is moved automatically to the test (e.g., Lazareva & Wasserman, 2010). In these cases, the accuracy at the end of the training could be used at terminal performance. We do not recommend using accuracy to the training trials during testing as terminal performance as the introduction of novel testing pairs can sometimes lead to decrement in accuracy to training pairs (e.g., Lazareva et al., 2004).

4. **Trying different ranges of free-fitting parameters and/or rerunning simulation** to ensure that you found the best possible solution. Section 3.2 provides some recommendations on selecting the ranges of free-fitting parameters.

5. **Setting reasonable standards for goodness-of-fit.** Generally, we consider a least-squared error under 2-3% (less than 0.02-0.03) to be a very good fit for behavioral data where this kind of variation can be ascribed to a measuring error (e.g., changes in subject's motivation, fatigue, and other environmental variables). In addition to the least-squared error, we advise to pay attention to the residuals for the *training pairs* (but not for the testing pairs). In a good, unbiased fit, the residuals are expected to have a sum that is close to zero. The biased fit often occurs because the model severely overpredicts (or underpredicts) a single pair while closely matching accuracy in all other training pairs. An unbiased fit with a slightly larger least-squared error is preferable to a biased fit with a smaller least-squared error.

2.0 GETTING STARTED

2.1. SYSTEM REQUIREMENTS

The compiled version of the TrI Toolbox is currently available for Windows 7 and 8. Contact Olga Lazareva at olga.lazareva@drake.edu if you are interested in obtaining the compiled version for a different platform.

The MATLAB system requirements are available at http://www.mathworks.com/products/matlab/index.html. The TrI Toolbox does not require installation of any other commercial or third-party toolboxes.

2.2. TOOLBOX INSTALLATION

All files are available at http://www.copal-lab.com/tri-toolbox. Download the toolbox version of the program, unzip the folder and save it to a desired location on your hard drive. Open MATLAB and either add the TrI Toolbox folder to your Matlab path or make the TrI Toolbox folder your current directory. Type *triToolbox* in the MATLAB command window and press Enter to start the program.

2.3. STANDALONE INSTALLATION

All files are available at http://www.copal-lab.com/tri-toolbox. If you are installing the TrI Toolbox for the first time, then select the full standalone version that includes MCR (MATLAB component runtime). The MCR is a free package provided by MATLAB that is required to run the compiled standalone executable and it requires about 200 MB of space on your hard drive.

If MRC (version 7.15) is already installed on your computer, then select standalone executable version with no MCR and skip Step 1 below.

Once you have downloaded the TrI Toolbox, unzip the folder and save it to the location where you want TrI Toolbox to be installed (for example, *C:\\Program Files\TrI Toolbox*). You can do that by double-clicking on the *.zip* folder and dragging *TrI Toolbox_1.0* folder into your preferred location. After that, you may delete *.zip* folder.

The *TrI_Toolbox_1.0* folder contains several files including readme.txt file, *triToolbox.exe* executable file, and (if you chose full standalone executable version) *MRCInstaller.exe*.

To complete installation:

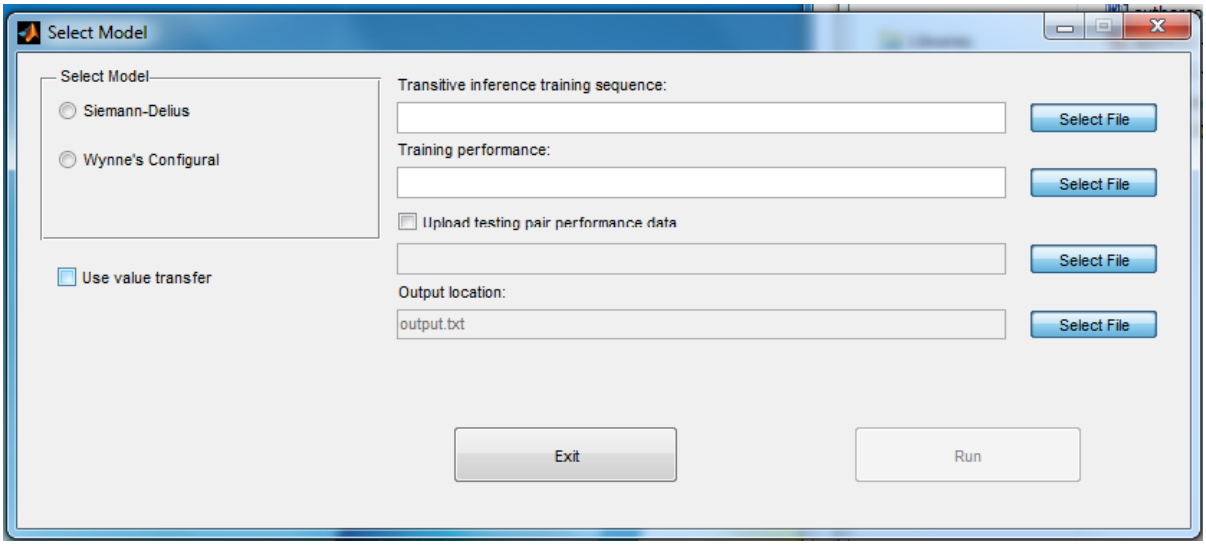
Step 1. (Full standalone version only) **Install MATLAB Component Runtime.** Double-click on *MCRInstaller.exe* and follow the instructions of the installation wizard.

Step 2. Start the TrI Toolbox. Double-click on *triToolbox.exe* to install/start the program.

3.0 HOW TO USE TrI TOOLBOX

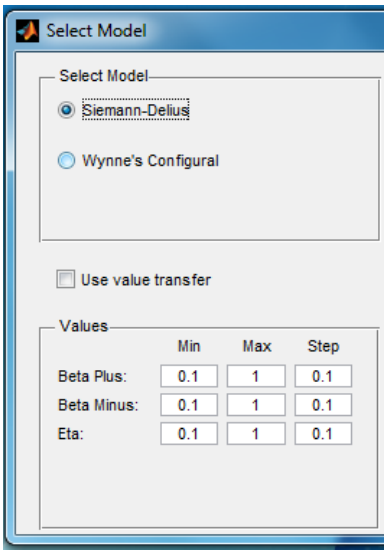
3.1. MAIN SCREEN AND PROGRAM CONTROL

This section describes the program's main screen (pictured below) which is fairly straightforward.



The options related to model selection and parameter setup are presented on the left, while the buttons related to file selection/saving are located on the right. The Run button is not available until the model is selected and the required files are uploaded.

3.1.1. Select Model options



Select a model that you want to apply to your data. The ranges of free-fitting parameters for the selected model in a separate window on the bottom left of the screen. The screenshot on the left shows this window for Siemann-Delius model.

If desired, a value transfer option can be added to either model by checking the option *Use value transfer*. This will add a fourth line under the model's parameters that starts with Delta (a free-fitting parameter determining the proportion of the transferred value). The currently implemented version assumes a bidirectional value transfer (both from a reinforced stimulus to a nonreinforced stimulus and vice versa) that occurs at the equivalent rate (determined by a single δ parameter).

Section 3.2 lists appropriate ranges of the free-fitting parameters for both models and provides recommendations for the model choice and selection of the parameter ranges. The *Step* option determines the increment with which the range of parameters will be sampled. For example, if the minimum value is 0.1 and the step is 0.1, then the sampled

parameter values will include 0.1, 0.2, 0.3, and so on until the maximum value is reached. If the difference between maximum and minimum value is not evenly divisible (e.g., min = 0.1, max = 0.95, and step = 0.1), then the maximum value will not be reached. Instead, the largest value that is evenly divisible by the increment will be tested (in our case, 0.9).

Important note: Be sure to change *Step* value as you are changing the range of the parameters. Forgetting this part can lead to undesirable results: For example, setting the minimum value at 0.01 and the maximum value at 0.1 but leaving the step at 0.1 means that the program will only test 0.01 (the minimum value) for this parameter.

3.1.2. Open/Save File options

On the right side of the main screen, two *Select File* buttons are given for opening the data file with the sequence of training trials (*training sequence file)* and the data file with the asymptotic performance for all training pairs (*training performance file)*. An optional data file with the performance to the testing pairs can be added if the option *Upload testing pair performance data* is checked. If this optional file is selected, then the output file will contain the subject's testing performance, the testing performance predicted by a model, and residuals (the difference between training performance and testing performance; see Section 3.4. for more details).

The last *Select File* button determines the location of the output file. By default, the output file is saved as *output.txt* in the same location as the executable (*.exe*) file. If the file with the same name already exists in the directory, you will see a warning that the output file will be overwritten. To avoid that you can change the default name and/or location of the file.

The required format of all input files is described in more details in Section 3.3. The output file is described in Section 3.4.

3.1.3. Running simulations

To run simulations, click on *Run* button in the bottom right corner of the main screen. This button is greyed out until the desired model is selected and the two required input files are selected. Once you click on *Run*, a progress bar will appear indicating the percentage of calculations performed. The calculation time depends on your hardware, the length of the data file, the range and the step of the parameters, and other factors. If the calculations take too long, then try closing other applications and/or reducing the range of the parameters to speed up the computation. Alternatively, take this time to drink a cup of coffee or go to lunch.

In some rare cases, you might encounter *Out of memory* error during simulations. Section 4.0 provides some ideas for remedying this issue.

3.2. CHOICE OF THE MODEL AND THE RANGE OF THE PARAMETERS

Table 1 lists the upper and lower ranges of the free-fitting parameters for each model, together with the ranges that, in our experience, produce good fit to pigeon, rhesus monkey, and human data. Consulting these recommendations might be useful if you have a difficult time locating an

acceptable solution. We do not have range recommendation for δ because in our experience addition of value transfer component does not improve the predictive ability of the model and is therefore unnecessary (see, for example, Lazareva et al., 2004; Lazareva & Wasserman, 2006).

Important note: The upper ranges of the several parameters (namely β, β_+, β_-, γ, and ε) should never exceed 1. The upper range of δ (a parameter specifying proportion of transferred value in a value transfer add-on option should be less than 0.5. There are very good theoretical and/or computational reasons for these restrictions; interested readers should consult relevant publications (Siemann & Delius, 1998; von Fersen, Wynne, Delius, & Staddon, 1991; Wynne, 1995).

Table 1. The upper and lower ranges of the free-fitting parameters for each model, together with the ranges that produced good fits to pigeon, rhesus monkey, and human data in our previous research.

	Possible ranges		*Pigeons*			*Rhesus monkeys*			*People*		
	Min	Max	Median	Min	Max	Median	Min	Max	Median	Min	Max
					Wynne model						
β	>0	1	0.003	0.0000001	0.99	0.99	0.07	1	0.95	0.22	1
α	1	∞	48	11	200	17	5	141	12	5	40
γ	>0	1	0.11	0.01	0.97	0.7	0.1	1	0.89	0.25	0.99
					Siemann-Delius model						
β_+	>0	1	0.03	0.004	0.03	0.02	0.002	0.11	0.1	0.06	0.99
β_-	>0	1	0.004	0.00001	0.1	0.02	0.00001	0.53	0.06	0.00001	0.99
ε	>0	1	0.54	0.01	0.99	0.2	0.00001	0.9	0.05	0.000001	0.4
					Value transfer add-on						
δ	>0	>0.5	--	--	--	--	--	--	--	--	--

Note that the ranges provided in Table 1 are based on the limited data set that was available to us in our prior research and should be taken as a guidance, not as the definitive word. If the model fails to provide a satisfactory fit, it is always a good idea to explore additional, unusual ranges of free-fitting parameters.

In our experience, Wynne's model tends to produce a slightly better fit than Siemann-Delius model although we currently have no formal, published comparisons to support this recommendation.

3.3. DATA FORMATTING

The program requires two data files, one with the sequence of training trials (*training sequence file)* and one with the asymptotic performance for all training pairs (*training performance file)*. The files must be saved in .csv (comma-separated format).

Training sequence file must be formatted as follows:

1. **The training pairs must be coded numerically in alphabetical order.** For example, the pair A+ B- must be coded as 1, the pair B+ C- must be coded as 2, and so on. The number of training pairs is unlimited.

2. **Accuracy must be coded numerically** as 1 (correct) and 0 (incorrect).

3. **The first line must indicate which data columns contain relevant information.** The first number specifies the column that contains numerical codes for the training pairs, while the second number indicates the column that contains accuracy codes. The third number is a reserved formatting character that is currently always zero. For example, the first line *4,7,0* indicates that the pair codes are in column 4 and the accuracy is in column 7.

Training performance file must contain asymptotic accuracy to each training pair *on a separate line* and *in alphabetical order*. For example, the file for the typical 5-term task will have 4 lines:

0.9
0.89
0.78
1.0

where the top number (0.9) indicates the asymptotic accuracy to the pair A+ B-, the next number (0.89) indicates the asymptotic accuracy to the pair B+ C-, and so on. The number of the lines in this file must be equal to the number of numerical pair codes in training sequence file. In other words, if your training sequence file contains pair codes 1 to 7, then your training performance file must provide 7 lines with the asymptotic accuracy.

The third, optional data file is the *testing performance file.* This file must contain alphabetically coded testing pairs, followed by subject's accuracy:

CE, 1.0
BD, 0.95
FH, 0.7

The alphabetical codes are not case-sensitive (in other words, BD, bd, and Bd are treated equivalently). The pairs can be listed in any order. If only a subset of all possible testing pairs is presented, then the output file will contain predictions for this subset and not for the other testing pairs.

3.4. Saving results and interpreting output file

The program saves the results of the simulations in a comma-separated text file. The header of the output file contains information about the location of the training sequence file and training performance file which could be useful if you wanted to check that you have uploaded the correct dataset. The third line contains the location of the output file. Finally, the last line in

header contains information about the selected model. Again, this could be useful if you wanted to make sure that you indeed selected the correct model.

The next heading, *Variables*, provides information about free-fitting parameters. The initial lines show the selected ranges and steps for each of the parameters. For example,

```
Beta , 0.10000:1.0000,0.10000
```

means that the parameter β was set with the minimum of 0.1, the maximum of 1.0, and it was tested in increments of 0.1.

After listing the ranges and the steps for all free-fitting parameters, the output file provides the best-fitting values of each parameter and the least-squared error.

The next heading, *Direct Values*, is followed by a list of elemental associative values for each training stimulus calculated using the best-fitting values of each parameter listed above, while the heading *Configural Values* precedes a list of configural associative values (where V(B|AB) indicates configural value of B in the pair AB, V(B|BC) indicates configural value of B in the pair BC, and so on).

The next heading, *Training Accuracy: Predicted-Actual-Residual*, is followed by the list of the accuracy to the training pairs calculated by the model using the best-fitting values of each parameter listed above; subject's accuracy to the training pairs provided in training performance file; and the residuals (where negative values indicate that the model underpredicted the actual performance and the positive values indicate that the model overpredicted the actual performance).

The final heading, *Testing Accuracy: Predicted-Actual-Residual*, is followed by the list of accuracy to all testing pairs predicted by the model on the basis of associative values listed above. If the testing performance file is uploaded, then this list also contains subject's accuracy to the testing pairs and the residuals; otherwise, these entries are left blank. Keep in mind that accuracy to the testing pairs is predicted rather than fitted (see Section 1.2 for more details). If the testing performance file contained a subset of all possible testing pairs, then the output file will contain predictions for this subset only.

3.5. USING MATLAB TO RUN TRI TOOLBOX

Typing *triToolbox* in the MATLAB command window and pressing Enter will start the GUI interface described in Section 3.1. The file *triToolbox.m* contains the GUI-related functions, while the file *modelmain.m* contains the functions related to computations and input/output. Both files have comments that should help the interested user to understand and/or modify the code.

Deploying TrI Toolbox in MATLAB environment gives a user an opportunity to customize the code to suit the needs of a specific project (e.g., visualizing the change in associative values of the training stimuli in course of a simulation or adapting the code for nonstandard training procedures). Please contact Olga Lazareva (olga.lazareva@drake.edu) if you have questions

about the code; note, though, that familiarity with basic MATLAB functions and / or interface is expected.

4.0. TROUBLESHOOTING

If you are running TrI Toolbox in MATLAB environment, the error message will appear in MATLAB command window. If you are running a standalone toolbox, then the error message is recorded in *error.txt* file located in the same directory as the standalone executable.

The error messages are listed below:

Message	Cause	Solution
Errors related to the formatting of the training sequence file		
`Improperly formatted training record file header`	The first line of the training sequence file is not formatted correctly	See Section 3.3 on how to format the training sequence file
`Specified column outside valid range`	The column number in the first line of the training file is larger than the number of columns available	Correct the first line of the training sequence file
`Invalid numerical input in specified column`	A nonnumeric value appears in a specified column	Correct the coding in the training sequence file
`Invalid binary response value: 0 or 1 expected`	Subject response is not coded as either 0 or 1	Correct the coding in subject response column of the training sequence file
Errors related to the formatting of the training performance file		
`Improper number of lines in training performance file`	The number of lines (i.e., accuracy values) in the training performance file does not match the number of stimulus pairs in the training sequence file	Make sure that the training performance file contains terminal accuracy for all training pairs (see Section 3.3 for more details)
`Improperly formatted training performance file`	Training performance file is not formatted correctly	Refer to Section 3.3 on how to format the training performance file
`Training performance values outside valid range`	Accuracy in the training performance file is outside of 0 <= accuracy <= 1 range	Format accuracy as proportion of correct choices (not as percent correct)
Errors related to the formatting of the testing performance file		
`Invalid format`	The testing performance file is not formatted correctly	Refer to Section 3.3 on how to format the testing performance file

`Invalid stimulus pair format`	A two-character pair code is expected	Correct the coding of the testing performance file
`Values outside valid range`	Accuracy in the training performance file is outside of 0 <= accuracy <= 1 range	Format accuracy as proportion of correct choices (not as percent correct)

In rare cases, a user may encounter an *out of memory* error. This may happen if you are running simulation with a wide range of free-fitting parameters and a small step. Although the exact point at which you are likely to run into a problem will depend on your specific hardware and software configuration, more than 200 values to-be-tested per each free-fitting parameter will likely to lead to an out-of-memory problem on an average computer. For example, Siemann-Delius model with a minimum of 0.001, a maximum of 1, and a step of 0.005 for each parameter is likely to produce an error.

Since out-of-memory error does not appear to be a frequent issue at the moment, we have decided to prioritize computing speed over memory handling. If you are running into this error, then you may want to try the following:

1) Test a wide range of parameters with a large step first, then explore the best-fitting region in more details. For example, you may first run Siemann-Delius model with a minimum of 0.01, a maximum of 1, and a step of 0.01 for each parameter. If this produces the best fitting results with, for example, a β_+ of 0.25, then set this parameter for the next simulation to the minimum of 0.1, a maximum of 0.3, and the step of 0.001 to fine-tune your results.

2) Close the other applications that may be consuming the memory during simulations.

If all else fails, contact Olga Lazareva at olga.lazareva@drake.edu: We may be able to modify the source code to improve memory handling if this turns out to be a serious issue for some users.

5.0 GETTING HELP

If you have a problem that is not addressed in this manual, then feel free to email to Olga Lazareva at olga.lazareva@drake.edu.

6.0 HISTORY AND TERMS OF USE

TrI Toolbox has been developed by David Goodman and Olga Lazareva of Psychology Department at Drake University (copyright 2014). This program is free software: you can redistribute it and/or modify it under the terms of the MIT license.

You should have received a copy of the MIT license along with this program. If not, see http://www.gnu.org/licenses/.

7.0 REFERENCES

Johnson-Laird, P. N. (1999). Deductive reasoning. *Annual Reviews in Psychology, 50*, 109-135.
Lazareva, O. F. (2012). Transitive inference in nonhuman animals. In T. R. Zentall & E. A. Wasserman (Eds.), *The Oxford Handbook of Comparative Cognition* (pp. 718-735). New York: Oxford University Press.
Lazareva, O. F., Smirnova, A. A., Bagozkaja, M. S., Zorina, Z. A., Rayevsky, V. V., & Wasserman, E. A. (2004). Transitive responding in hooded crows requires linearly ordered stimuli. *Journal of Experimental Analysis of Behavior, 82*, 1-19. doi: 10.1901/jeab.2004.82-1
Lazareva, O. F., & Wasserman, E. A. (2006). Effect of stimulus orderability and reinforcement history on transitive responding in pigeons. *Behavioural Processes, 72*, 161-172. doi: doi:10.1016/j.beproc.2006.01.008
Lazareva, O. F., & Wasserman, E. A. (2010). Nonverbal transitive inference: Effects of task and awareness on human performance. *Behavioural Processes, 83*, 99-112. doi: doi:10.1016/j.beproc.2009.11.002
Pearce, J. M., & Wilson, P. N. (1990). Configural associations in discrimination learning. *Journal of Experimental Psychology: Animal Behavior Processes, 16*, 250-261.
Siemann, M., & Delius, J. D. (1998). Algebraic learning and neural network models for transitive and non-transitive responding. *European Journal of Cognitive Psychology, 10*, 307-334. doi: 10.1080/713752279
Vasconcelos, M. (2008). Transitive inference in non-human animals: An empirical and theoretical analysis. *Behavioural Processes, 78*, 313-334. doi: 10.1016/j.beproc.2008.02.017
von Fersen, L., Wynne, C. D. L., Delius, J. D., & Staddon, J. E. R. (1991). Transitive inference formation in pigeons. *Journal of Experimental Psychology: Animal Behavior Processes, 17*, 334-341. doi: 10.1037/0097-7403.17.3.334
Wynne, C. D. L. (1995). Reinforcement accounts for transitive inference performance. *Animal Learning and Behavior, 23*, 207-217.
Wynne, C. D. L. (1997). Pigeon transitive inference: Tests of simple accounts of a complex performance. *Behavioural Processes, 39*, 95-112.
Wynne, C. D. L. (1998). A minimal model of transitive inference. In C. D. L. Wynne & J. E. R. Staddon (Eds.), *Models of action: Mechanisms for adaptive behavior* (pp. 269-306). New York: Lawrence Erlbaum Associates.

8.0 APPENDIX: OVERVIEW OF THE MODELS

The operating principle behind Wynne and Siemann-Delius models of TI is, at base, simply that the value of each training stimulus is related to the stimulus's prior association with reinforcement, increasing with every reinforcement and decreasing with every non-reinforcement. A total value of the stimulus is composed of two parts, a context-independent direct value and a context-dependent configural value (Pearce & Wilson, 1990). The direct value of a stimulus is modified each time the stimulus is presented, whereas the configural value of the stimulus is modified each time the stimulus is presented in a given pair. The choice of the stimulus in a training pair is determined by a weighted average of both values. In testing pairs, the configural value is equal to zero as the two stimuli, by definition, have not been presented together before; thus, subjects' behavior to the testing pairs is determined solely by the direct values of the stimuli.

8.1. WYNNE MODEL

The Wynne model (Wynne, 1998) is a modified version of the Rescorla-Wagner model. In this model, the direct values for the pair X+ Y- are updated as follows:

$$V(X)_{i+1} = V(X)_i + \beta * [1 - V(X)_i] \quad (1)$$

$$V(Y)_{i+1} = V(Y)_i - \beta * V(Y)_i \quad (2)$$

where $V(X)_i$ and $V(Y)_i$ are associative values of the stimuli X and Y accumulated at the end of trial *i*, $V(X)_{i+1}$ and $V(Y)_{i+1}$ are associative values of the same stimuli calculated for the next trial, $i + 1$, and β is a free-fitting parameter ranging from 0 to 1 and indicating speed of learning. Higher values of β will lead to a proportionately larger changes in associative values of

both stimuli and, consequently, to a faster improvement in accuracy, while lower values of β will be associated with smaller changes in associative values and slower learning rate.

The configural values of the two stimuli, $V(X|XY)$ and $V(Y|XY)$ are updated in the same fashion as direct values:

$$V(X|XY)_{i+1} = V(X|XY)_i + \beta * [1 - V(X|XY)_i] \quad (3)$$

$$V(Y|XY)_{i+1} = V(Y|XY)_i - \beta * V(Y|XY)_i \quad (4)$$

The probability of the subject selecting the stimulus X in the pair XY is given by a sigmoid function:

$$P(X|XY) = \frac{1}{1+e^{-\alpha(2r-1)}} \quad (5)$$

where α is a free-fitting parameter that determines the degree of sensitivity to the differences in associative values and the term *r* is calculated as:

$$r = \frac{V(X)+\gamma(V(X|XY)}{V(X)+V(Y)+\gamma[V(X|XY)+V(Y|XY)]} \quad (6)$$

The parameter γ is the third free-fitting parameter of the model that determines the degree to which choice behavior is influenced by the direct and configural values. The higher values of γ indicate stronger reliance on configural values while the lower values indicate their weaker influence.

8.2. SIEMANN-DELIUS MODEL

The Siemann-Delius model (Siemann & Delius, 1998) is a modified version of the Luce's model. In this model, the direct values V(X) and V(Y) in a pair X+ Y- are updated according to the following equations:

$$V(X)_{i+1} = V(X)_i + \beta_+ * V(X)_i * \varepsilon \quad (7)$$

$$V(Y)_{i+1} = V(Y)_i - \beta_- + V(Y)_i * \varepsilon \quad (8)$$

where β_+ and β_- are the free-fitting parameters that determine the speed of learning for reinforced and non-reinforced stimuli, varying from 0 to 1, and ε is a weighting parameter that determines the degree to which the direct values influence choice behavior in comparison to configural values. Higher values of ε indicate stronger influence of the direct values in comparison to the configural values (and vice versa).

The configural values are updated as follows:

$$V(X|XY)_{i+1} = V(X|XY)_i + \beta_+ * V(X|XY)_i\kappa \quad (9)$$

$$V(Y|XY)_{i+1} = V(Y|XY)_i - \beta_- * V(Y|XY)_i\kappa \quad (10)$$

where κ is a weighting parameter for configural values such that

$$\kappa = 1 - \varepsilon \quad (11)$$

The probability of the subject selecting the stimulus X in the pair X+ Y- is calculated as follows:

$$P(X|XY) = \frac{V(X)*V(X|XY)}{V(X)*V(X|XY)+V(Y)*V(Y|XY)} \quad (12)$$

8.3. VALUE TRANSFER ADD-ON

While reinforcement-based models described above simply posit that the ordered series of associative values arise as a result of specific training procedure, value transfer theory postulates the bidirectional transfer of associative value between the reinforced and non-reinforced stimuli (von Fersen et al., 1991). According to the value transfer theory, the associative value of a stimulus Y that occurs in a pair X+ Y- and Y+ Z- is composed of its own value accrued independently, a positive value transferred from the stimulus X, and a negative value transferred from the stimulus Z according to the following equations:

$$V(X)_{i+1} = V(X)_i - \delta(V(X)_i - V(Y)_i) \quad (13)$$

$$V(Y)_{i+1} = V(Y)_i + \delta(V(X)_i - V(Y)_i) \quad (14)$$

where δ is a free-fitting parameter determining a proportion of transferred value such that $0 < \delta < 0.5$.